A3

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/24 — A2

(11) International Publication Number: WO 00/52570

(43) International Publication Date: 8 September 2000 (08.09.00)

(21) International Application Number: PCT/US00/05674

(22) International Filing Date: 2 March 2000 (02.03.00)

(30) Priority Data:

60/122,591	3 March 1999 (03.03.99)	US
60/122,594	3 March 1999 (03.03.99)	US
60/122,595	3 March 1999 (03.03.99)	US

(71) Applicant: CIVON CORPORATION [US/US]; 2336H Walsh Avenue, Santa Clara, CA 95051 (US).

(72) Inventors: PANG, Joseph; 2336H Walsh Avenue, Santa Clara, CA 95051 (US). TENG, Peter; 2336H Walsh Avenue, Santa Clara, CA 95051 (US).

(74) Agents: TITUS, Carol, D. et al.; Leary & Associates, 505 W. Olive Avenue, Suite 330, Sunnyvale, CA 94086 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

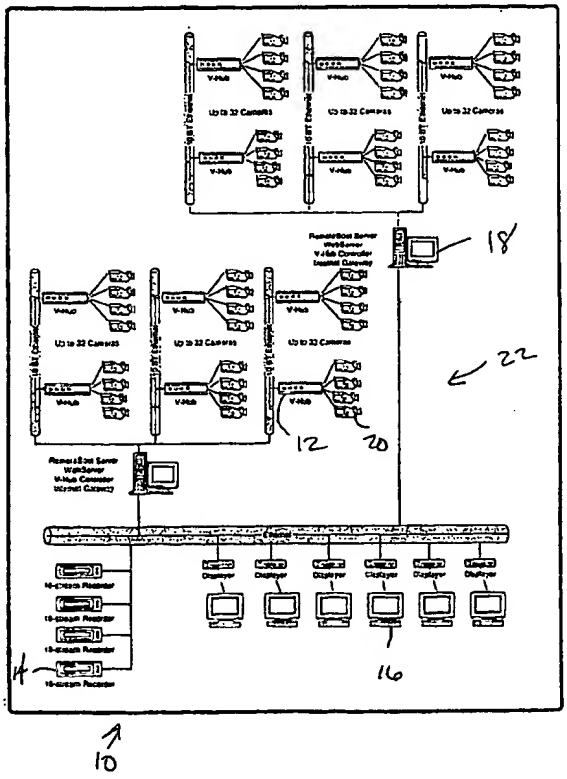
Published

Without international search report and to be republished upon receipt of that report.

(54) Title: A NETWORKED CLIENT/SERVER EMBEDDED BACKBONE SYSTEM INTEGRATING OPERATING SYSTEM REMOTE-BOOTING, APPLICATION SOFTWARE PUSHING, AND CENTRAL CONTROL COMMUNICATION SOFTWARE

(57) Abstract

A networked embedded backbone system integrates three techniques to make the system better than traditional standalone EPROM-based embedded systems. The first technique is remote booting of operating system onto the embedded systems from a central server. Once the operating system is booted, the application software on the embedded systems may then be pushed from a central server. A generic central control communication software layer is provided for application software on each embedded system to communicate with the central server or other embedded modules on the network.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Utility Patent Application

of

Joseph Pang and Peter Teng

for

A NETWORKED CLIENT/SERVER EMBEDDED BACKBONE SYSTEM INTEGRATING OPERATING SYSTEM REMOTE-BOOTING, APPLICATION SOFTWARE PUSHING, AND CENTRAL CONTROL COMMUNICATION SOFTWARE

FIELD OF THE INVENTION

This application relates to a network of embedded systems that are controlled by a central server. Optional encryption and motion detection in H.263 compressed video stream systems are also disclosed. This application relates to motion detection in video surveillance equipment. More particularly, it relates to a more efficient method of motion detection in H.263 compressed video stream.

BACKGROUND

A traditional computer embedded system is a standalone computer system with its operating system and application software implemented in a PROM (Programmable-Read-Only-Memory). Embedded systems are devices with small computerized controllers which work in the everyday world such as thermostats, video surveillance cameras, automotive self diagnostics, robotic control for automated manufacturing, traffic lights and other household, industrial and other applications. To change or update the function or behavior of the embedded system requires its PROM be replaced with a new one manually. Standalone embedded systems are not easily controlled by a central server, and generally provide little or no flexibility and scalability of the system without complete replacement.

SUMMARY OF THE INVENTION

To overcome the difficulties of utilizing such standalone embedded systems, a networked client/server embedded backbone system is designed. The new networked embedded system has several benefits.

The first benefit is financial savings. Each embedded system boots its Operating System (OS) into its RAM (Random-Access-Memory) from the central server instead of running from its PROM. It's application software is also pushed from the central server into the RAM after the OS is booted. There is a great cost saving because a RAM device is much lower in cost than the corresponding PROM device.

The networked embedded system is also easy to use. Both the OS and application software are transferred from the central server when the embedded system is powered up. There is no need to manually change the PROM. Function modification and updating can be easily managed from the central server, thereby alleviating the need to update each embedded system manually.

The networked embedded system is flexible to allow the user to utilize a wide variety of options. For example, a RAM device's larger capacity makes the selection of OS more flexible and application software easier to implement. A traditional standalone embedded system requires a small footprint of the OS, so the choices of OS are very limited. With the new design, a larger size OS such as Linux or Windows can be used.

The networked embedded system is scalable. Traditional standalone embedded systems are severely limited by their PROM size. The application software pushing in the new design requires only a larger size of RAM so it is much easier to implement more complicated application software.

The networked embedded system is reliable. Diskless OS remote booting and software pushing do not require any mechanical parts on the embedded system. It is as reliable as the traditional standalone embedded system.

The networked embedded system is easily managed by the user. In addition to transferring OS and application software from the central server to the embedded systems, a central control communication software layer is provided to make a large number of embedded systems possible and more manageable.

The present invention takes the form of a networked embedded backbone system that integrates three techniques to make the system better than traditional standalone

EPROM-based embedded systems. The first technique is remote booting of the operating system onto the embedded systems from a central server. Once the operating system is booted, the application software for the embedded systems may then be pushed from a central server. A generic central control communication software layer is provided for application software on each embedded system to communicate with the central server or other embedded modules on the network.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows the components of the present invention used a digital video surveillance system.
Figure 2 shows an overview of the full system.

DETAILED DESCRIPTION

Three techniques are integrated together to make the networked embedded backbone system better than traditional standalone EPROM-based embedded systems. The first technique is remote booting of operating system onto the embedded systems from a central server. Depending on the targeted operating system to be booted onto the embedded system, a small footprint of the boot ROM is programmed and installed on the embedded system motherboard or Network Interface Card (NIC). This small boot ROM program will use the BOOTP and TFTP (Trivial File Transfer Protocol) protocols to locate a boot server or central server on the network and request the loading of the boot image file from that server.

On the server side, a boot server code is implemented to answer the embedded systems' request and to provide booting services. The boot server transfers a designed boot image file based on each requester's unique MAC (Media Access Control) address on the network interface card. After the boot image file is transferred onto the embedded system and executed, the final footprint of the targeted OS is then transferred and executed from the boot server to the embedded system.

A different OS can be booted to different embedded systems based on requester's MAC address. This process is different for different types of OS and can go through a number of phases before the fully functional OS is booted onto the embedded system.

Once the operating system is booted, the application software on the embedded systems may then be pushed from a central server. Since different OS image files can be booted to the embedded systems depending on requester's unique MAC address, the image of the

targeted OS can be setup with a special initialization process so during its execution phase, a different pre-packaged application software can be loaded. An embedded system usually handles some unique devices on that system; specially packaged application software for that system can be loaded and run after the OS is booted.

A generic central control communication software layer is provided in the packaged application software for application software on each embedded system to communicate with the central server or other embedded modules on the network. It provides communication and control mechanisms among the central server application software and the embedded system application software, and thereby provides several services.

The first service is advertisement, which allows an individual embedded system to advertise its existence after its power-up. A method of periodical multicast of data gram packets is used to allow each embedded system to advertise its existence and search for a central controlling device.

A capability information exchange service allows the central server to know the capability information of every embedded system on the network.

A RPC (Remote Procedure Call) service allows the central server to communicate and control every individual embedded system on the network. It also allows each embedded system to communicate and control other embedded systems on the network.

EXAMPLE EMBODIMENT

A digital surveillance system is implemented according to the methods described in this patent application and shown in figures 1 and 2. Figure 1 shows the components of the present invention used as a digital video surveillance system 10. Figure 2 shows an overview of the full system.

In this implementation, the V-Hubs 12, recorders 14 and displays 16 are "Networked Embedded Systems". The server 18 is the central server, which, in this case, serves as the remote-boot server as well as the central controller. Hundreds of cameras 20 may be connected to the V-Hub 12, devices that in turn are connected to the Ethernet network 22.

Linux OS and the V-Hub application are remote booted from the central server 18 and the V-Hub application will capture each camera's video streams and multicast them onto the network 22. The V-Hub 12 embedded system has a communication/ control software layer implemented so that it will report its existence to the central

controller 18 and accept capturing commands from the central controller 18 for execution.

Linux OS and the recorder application are remote-booted from the central server 18 and the recorder application will record video streams according to the pre-programmed schedule. The recorder 14 embedded system has a communication/ control software layer implemented so that it will report its existence to the central controller 18 and accept recording schedule or commands from the central controller 18 for execution.

Linux OS and the display application are remote-booted from the central server 18 and the display application will display any pre-programmed video streams. The display embedded system has a communication/control software layer implemented so that it will report its existence to the central controller 18and accept displaying schedule or commands from the central controller 18 for execution.

In this example, the whole "Networked Embedded System" forms a complete digital video surveillance system 10.

If desired an encryption system may be utilized to secure the transmission. An example of such an encyption system relies on the correct delivery of a secret key triple to all recipients. A centralized key distribution server is suitable for this purpose. Secure transfer of the key triple can be accomplished by any one of the standard mechanisms as described by Steiner et al. (J. G. Steiner, B. Clifford Neuman, and J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems", Proceedings of the Winder 1988 Usenix Workshop on Workstation Security, Portland, OR, August, 1988), Garfinkel (Simson Garfinkel, PGP: Pretty Good Privacy, O'Reilly & Associates, 1994), and US Patent Number 5,214,706 to Minde (additional information found at http://www.ascom.ch/infosec), which is hereby incorporated by reference in its entirety, including any public key authentication and encryption techniques, or any other suitable technology. The key triple is determined by the multimedia data source and registered in a secure way to the key server. At any time, the key triple can be updated by the data source. A recipient can request the key triple from the server at any time and use the key triple to decode any multimedia data messages. The key triple is therefore time-variant.

The present method also relies on a pseudo random number generator (e.g. see Knuth, Donald, The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd edition, Addison-Wesley, 1998), that is known to the data source and all recipients. One could employ a pseudo number generator with pre-defined

parameters, or one could rely on the key server to distribute that information to all recipients. The parameters that characterize the generator need not be kept secret. Yet, the generator must indeed generate pseudo random numbers with a very large period. Furthermore, the sequence must be reproduced in a deterministic manner.

Let us denote the pseudo random generator by $G(\sum)$. Thus, if the current random number is k, then the next number in the sequence in $G(k)$. The application of the generator to k for n times is denoted by $G^n(k)$. The relative positional index of $G^n(k)$ with respect to k is n. The key triple is defined by (a,b,p) where a is the fixed index key, b is a random number within the pseudo random number sequence, and p is the relative index of b with respect to the initial random number. The initial selection of the triple is done at the data source. a is chosen according to the selected strong cipher mechanism (e.g., see Minde) and kept fixed throughout the session. b can be generated randomly as long as it belongs to the pseudo random sequence defined by $G(\sum)$. The initial value for p is zero.

The multicast multimedia data is grouped into variable length messages, each encrypted by a variable key. Denoted by XOR(v,u), the encryption of a message (i.e., a byte sequence) v using a key u is done by laying out the bytes of u periodically into an infinite sequence of bytes and applying it against v using the XOR operation. Decryption is done easily by the same operation. Each message is preceded by an encrypted index. The index must be encrypted by a strong cipher mechanism such as that described in Minde. We shall denote the encrypted index by $Z(i)$ where i is the unencrypted index.

The multicast source encrypts the message by first selecting an initial random number and setting the initial index to zero. Then an incremental index D is selected randomly from the integer set [0,1,...MaxIndex] according to a uniform distribution. The previous random number is passed through $G(\sum)$ for D number of times to yield the current random number. This current random number will be the key used to encode the current message. The absolute index, that is, the accumulation of all the incremental indices, will be encrypted using the strong cipher $Z(\sum)$. For transmission, the encrypted index and the encrypted message is concatenated and transmitted.

Once a message has been received, it is decrypted. The absolute index is first decrypted using the inverse of $Z(\sum)$, and then the random number used to encrypt the message is calculated. This calculation can be done quickly from the last known index and random key pair. This applies even when there is message loss. In the worst case where a receiver may have lost all the transmitted messages (e.g., a receiver happens to start the reception when the session is already in progress), the receiver can compute

the current random key by passing the initial random key through $G(\sum)$ according to the absolute index. Unfortunately, the absolute index can be very large and require a lot of computations. To reduce this computation, it is recommended that the multicast source update the current index and random key pair on the key distribution server periodically. Thus, receivers that have lost the encryption key can always obtain the last registered key triple from the server and limit the computations needed to obtain the current key.

If desired a motion detection system may be utilized to determine whether an alarm, a recording or other action is necessary. An example of a motion detection system used for H.263 bitstream syntax is based on the Discrete Cosine Transform (DCT) compression with motion estimation and prediction. Pictures are partitioned into a number of macro-blocks (MB). Some pictures are encoded in the INTRA mode but the majority in INTER mode. In INTRA mode, a picture is encoded without prediction and motion estimation. This serves as an anchor for decoding subsequent INTER coded frames. For INTER coded pictures, motion estimation and prediction are used. For motion estimation, each MB is compared against the previous picture within a search range centered around the current MB to find the best match. Once the best match is found, the difference between the current MB and the best match is computed. This difference is transformed into the frequency domain using DCT. The spectral components of the DCT output are quantized and encoded using Huffman or Arithmetic encoding. The resulting bits from each MB compression are packed and wrapped with header information to form the data stream of one compressed picture. Multiple compressed picture streams are concatenated to form a typical H.263 encoded bit stream.

For a motionless scene, all INTER picture will have all MBs unencoded, resulting in a minimum number of bits per picture. In fact, in a perfect environment with no noise, this is a necessary and sufficient condition for non-motion. In reality, the capture and encoding processes are subject to noise. Environmental noise such as changes of sunlight, small vibrations, changes in temperature, etc., can all contribute to perceived motion by the camera. Furthermore, the analog-to-digital capture process introduces A/D noise. Finally, one major source of noise is in the compression process, where the DCT coefficients are quantized according to a given quantizer to achieve high ratio but lossy compression.

The motion detection method described below is based on empirical observations that even though there are different sources of noise that would create

deviation from the ideal scenario, the noise disturbance remains a perturbation from the ideal case.

The method utilizes the H.263 bitstream syntax as described in "Video Coding for Low Bitrate Communication." For an INTRA coded picture, no motion detection is performed. For an INTER coded picture, the bit stream is parsed to obtain the following values:

I = number of INTRA coded MB
M = number of INTER coded MB with non-zero motion vector
M'= number of INTER coded MB with zero motion vector
K' = number of non-coded MB
T = total number of MB

To obtain these values, one must parse the whole picture layer bitstream. However, the inverse quantization, inverse DCT and image reconstruction can be skipped. By skipping the unnecessary processing, the computation burden is greatly reduced.

Once the above key values are obtained, we use a number of successive criteria to determine whether motion has taken place. Ideally, there is no motion if and only if I=M=M'=0 and K'=T. However, due to noise disturbance, we must modify the criteria to deal with non-ideal situations. The criteria are based on a number of user-defined thresholds.

First, if $I/T > 1-a$, a declaration of non-motion is given because there are not enough representation of motion estimated MBs. a is preferably a small positive number. The smaller the number, the lower the tolerance for motion. If the first criterion is not met, then if $M/(M+M'+K) > b$, then a declaration of motion is given because there is sufficient non-zero motion vectors to indicate motion. b should be a fairly small positive number. If the second criterion is not met, then the average bits used to encode a macro-block in this picture is computed, and if this value exceeds 1+g, then a declaration of motion is given. The last criterion is designed to cope with encoders that are not very good at motion estimation. For these encoders, it is possible to have many MB coded with zero Motion Vector, but the total number of bits used to encode each MB must be correspondingly high if there is indeed motion. g can be any positive value.

CONCLUSION

The uniqueness of the Networked Client/Server Embedded Backbone System is a combination of its networking feature, operating system remote booting feature, application software pushing feature, central control and communication software feature, central manageability, flexibility, ease of use and scalability.

The networked embedded Backbone System is different from other embedded system by its integration of three methods: remote booting of a highly sophisticated operating system from a central server instead of utilizing an EPROM-based function-limited OS, application software for the embedded system is pushed from a central server onto the RAM device of this system instead of being programmed on an EPROM device and a common layer of communication and control software is provide on each networked embedded system allowing inter-module controls from the central server to each individual embedded system.

Although the examples given include many specificities, they are intended as illustrative of only a few possible embodiments of the invention. Other embodiments and modifications will, no doubt, occur to those skilled in the art. Thus, the examples given should only be interpreted as illustrations of some of the preferred embodiments of the invention, and the full scope of the invention should be determined by the appended claims and their legal equivalents.

What is claimed is:

1. A method for creating and using a networked embedded backbone computer system, comprising the steps of:
   (a) booting over a network an operating system onto a plurality of embedded systems;
   (b) pushing selected application software onto said plurality of embedded systems from a central server;
   (c) and communicating between said plurality of embedded systems and said central server.

2. The method of claim 1 wherein step (a) is performed by booting from said central server.

3. The method of claim 1 wherein step (a) is performed by the steps of:
   (a) creating a footprint of a Boot ROM program within one of said embedded systems;
   (b) locating a boot server connected to said network;
   (c) requesting loading of the boot image filed from said boot server;
   (d) transferring a selected boot image file to said embedded system;
   (e) transferring a final footprint of a selected operating system;
   (f) and executing the selected operating system on said embedded system.

4. The method of claim 3 wherein said boot image file is selected based on a unique designation from said embedded system.

5. The method of claim 4 wherein said unique designation is a media access control address located on a network interface card.

6. The method of claim 3 wherein said Boot ROM program is created on a motherboard.

7. The method of claim 3 wherein said Boot ROM program is created on a network interface card.

8. The method of claim 1 wherein further communication occurs between said plurality of embedded systems.

9. The method of claim 1 wherein step (c) is performed using a central control communication software layer.

10. The method of claim 9 further comprising the step of utilizing said central control communication software layer to send a notification to said central server.

11. The method of claim 10 wherein said notification informs said central server of one of said plurality of embedded systems being ready to receive booting information.

12. The method of claim 10 wherein said notification includes information about capabilities of one of said embedded systems.

13. The method of claim 1 wherein said central server controls said plurality of embedded systems.

14. The method of claim 1 wherein said networked embedded backbone computer system is used as a digital surveillance system.

15. The method of claim 14 wherein at least one of the embedded systems is a surveillance camera and wherein said surveillance camera is monitoring a location.

16. The method of claim 15 further comprising the step of testing information received from said surveillance camera for motion.

17. The method of claim 16 wherein an alert is activated when motion is detected.

18. The method of claim 1 wherein the communications of step (c) are encrypted.

19. A method of encrypting data, comprising the steps of:

(a) selecting an initial random number;

(b) setting the initial index to zero;
(c) randomly selecting an incremental index from an integer set [0,1,...MaxIndex] according to a uniform distribution;
(d) passing the initial random number through $G(\sum)$ for a number of times equal to the incremental index to yield a current random number;
(e) and using the current random number as a key to encode a current message.

20. The method of claim 19 further comprising the steps of:
(f) repeating step (e) to create successive random number indices to encode successive messages.

21. The method of claim 20 further comprising the steps of:
(g) creating an absolute index from an accumulation of the initial number index and the successive random number indices;
(h) encrypting the absolute index using a strong cipher $Z(\sum)$;
(i) concatenating the encrypted index and the encrypted message;
(j) transmitting the encrypted index and the encrypted message.

22. The method of claim 21 further comprising the steps of:
(f) decrypting the absolute index using the inverse of $Z(\sum)$;
(g) calculating the random number used to encrypt the message;
(h) and decrypting the message.

23. A method for detecting motion in a bitstream having an H.263 syntax to return a state of motion or a state of non-motion, the method comprising the steps of:
(a) parsing the bitstream to obtain a set of values;
(b) obtaining an I value equal to a number of INTRA coded macro-blocks;
(c) obtaining an M value equal to a number of INTER coded macro-blocks with a non-zero motion vector;
(d) obtaining an M' value equal to a number of INTER coded macro-blocks with a zero motion vector;
(e) obtaining an K' value equal to a number of non-coded macro-blocks;
(f) obtaining an T value equal to a total number of macro-blocks;
(g) selecting a value for a;

(h) calculating I/T and 1-a;

(i) when I/T is greater than 1-a, declaring the state of non-motion.

24. The method of claim 23 further comprising the steps of:

(j) selecting a value for b;

(k) calculating M/(M+M'+K);

(l) when M/(M+M'+K) is greater than b, declaring the state of motion.

25. The method of claim 24 further comprising the steps of:

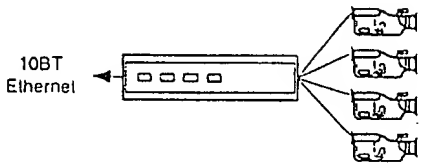
(m) selecting a value for g;

(n) calculating an average of bits used to encode the INTER coded macro-blocks;

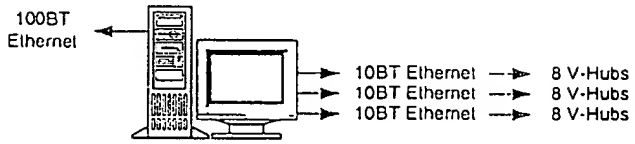
(o) when the average of bits is greater than 1+g, declaring the state of motion.

A Digital Video Surveillance System
System Components

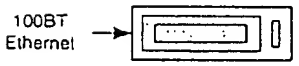
12
4-Camera V-Hub



18
RemoteBoot Server
WebServer
V-Hub Controller
Internet Gateway



14
16-stream Motion-Triggered
Recorder



16
4-Video Displayer

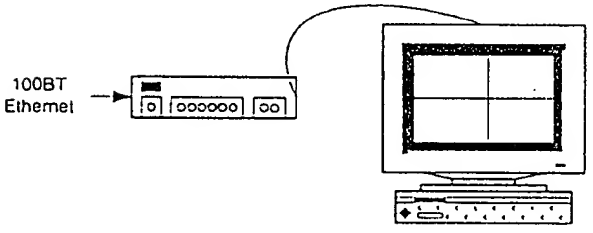


FIG. 1

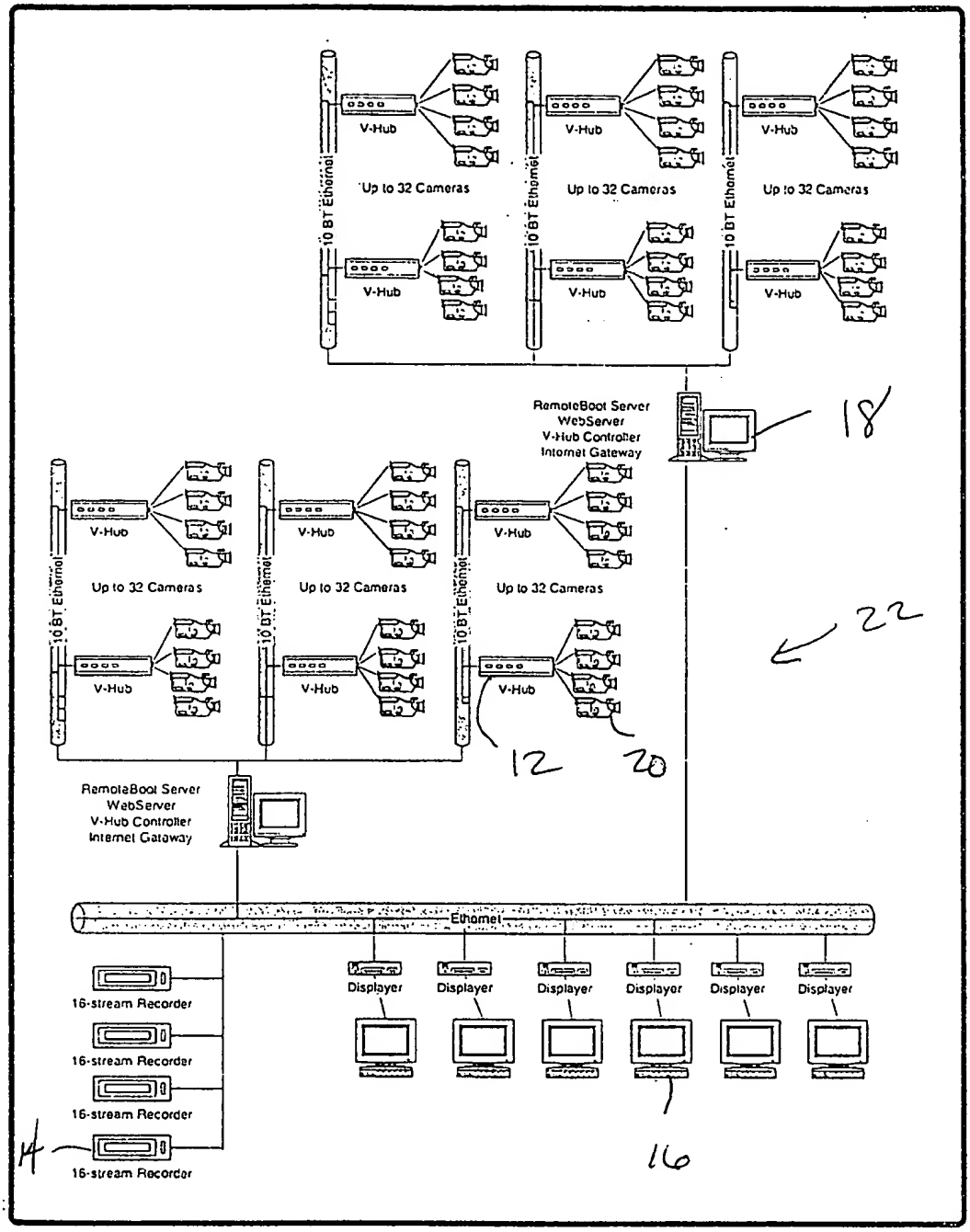

V-Hub
Up to 32 Cameras
10 BT Ethernet
RemoteBoot Server
WebServer
V-Hub Controller
Internet Gateway
18
22
12
20
Ethernet
Displayer
16-stream Recorder
14
16
10

FIG. 2